Computer Vision System with 2D and 3D Data Fusion for Detection of Possible Auxiliaries Routes in Stretches of Interdicted Roads

Diego Renan Bruno
University of São Paulo
diego.bruno@usp.br

Lucas P. Nunes Matias
University of São Paulo
lucas.matias@usp.br

Jean Amaro
University of São Paulo
jean.amaro@usp.br

Fernando Santos Osório
University of São Paulo
f.osorio@icmc.usp.br

Denis Wolf
University of São Paulo
denis@icmc.usp.br

Abstract

In this paper we present an intelligent system to help autonomous vehicles in real cities and with local traffic rules. A 2D and 3D visual attention system is proposed, capable of detecting the use of signs and aids in cases of major roadblock (road under work, with a traffic accident, etc.). For this to be possible, we analyze the cones and traffic signs that usually alert a driver about this type of problem. The main objective is to provide support for autonomous vehicles to be able to find an auxiliary route that is not previously mapped. For this we use a Grid Point Cloud Map. Using the ORB-SLAM visual odometry system we can correctly fit each stereo frame point cloud in the pose where the images were collected. With the concatenation of point clouds generated by the stereo camera, every grid block can draw the main characteristics of its region and an auxiliary route can be mapped. In this type of situation the vision system must work in real time. The results are promising and very satisfactory, we obtained an accuracy of 98.4% in the 2D classification task and 83% accuracy in the single frame 3D detection task.

1. Introduction

Autonomous car have been intensely studied since the last decade, more recently intelligent cities have been idealized and proposed methods, trying to spread the processing and decision making between intelligent vehicles and the smart city. The communication between an intelligent city and an autonomous car is based on the information exchange about the environment where the vehicle is inserted. The vehicle pass to the city its new collected information about the surroundings while the city send to the car the previous knowledge about the environment. With this mutual data swap, the vehicle can have a previous knowledge of the road and planned trajectory, and the city can update its stored information as new data is received from the intelligent vehicle.

In an ideal world, the city and the car have the map of the roads and safe routes can be easily calculated for the vehicle navigation. Yet, in some specific cases (urgent lane repair, broken car on the road and other eventual situations) the previously mapped road have to be temporarily obstructed. In these situations, the vehicle have an outdated map that can not be used for trajectory planning. The environment changes in an obstructed road with traffic signalization and cones have to be identified. Once this environment modification is noted, a vision system have to be activated and, if possible, a traversable auxiliary route have to be mapped, then, the vehicle can maintain its trajectory to the destination and notify the smart city data center about the temporarily road condition.

In this paper we propose a method of traffic signs and cone detection, and a grid point cloud mapping based on stereo 3D vision. Using the extracted object from the stereo point cloud, then we use a Multi-Layer Perceptron to recognize it, based on its 3D features. The cone features can be extracted from the road obstruction scene. Besides the cone detection we use Deep Learning to train a Convolutional Neural Network (CNN) for the case where a lane obstruction be informed by a traffic sign. Also, we present a Stereo Vision based Grid Mapping to identify possible auxiliary routes to recreate an alternative route to the planned destination.

In the next sections we gather related works, then we present our algorithm of traffic sign detection using 3D data, following we have the theoretical background and proposed 3D feature extraction and classification, thereafter the Deep Learning method to classify the type of traffic sign, then we present our 3D point cloud grid mapping, lastly we show our experimental results, conclusions and future works.

2. Related work

A great number of mapping methods propose a grid mapping based on 3D LiDAR sensor [1, 2]. A 2D grid is created and in each grid block is stored more detailed

URI: https://hdl.handle.net/10125/60174
ISBN: 978-0-9981331-2-6

information about the points in that region. With this approach we can reduce the number of data stored and maintain the information about the points in each grid block, becoming possible to identify the traversable area.

Some other mapping methods [3, 4] creates a top-view image with traversable area classified and represented in the image. The information about the environment is represented in the image by the pixel color (grid cells/blocks). Each pixel assumes a specific color, mapping the environment with the traversability information.

More recently new approaches focusing on a complete visualization of the surrounding have been proposed. In [5, 6] 3D information was collected and, using visual odometry the points collected are fused with the vehicle pose to recreate a 3D map of the environment. These points are used to create a mesh of the whole trajectory and the images collected within the 3D points can be used as texture for the mesh or to segment the obstacles and traversable areas. In order to visualize this mapping, methods can reconstruct the 3D representation close to the real scenario, but for a real time processing these mapping systems gather a volume of information that is not necessary for the vehicle navigation and would need too many processing power.

In the context of visual odometry, the methods proposed in [7, 8] can give a precise pose estimation based only in images captured during the executed trajectory. In each frame features are collected from the image and mapped with the 3D positioning. This process is repeated in the images captured during the trajectory, each time a feature collected is matched with one already mapped, the system uses the mapped position to estimate the motion of the vehicle and estimate a pose for the actual frame. This process can give a reliable pose estimation, using only images.

In a work done by Timofte and Zimmermann [9], a system was developed capable of detecting traffic signs using 3D data. In this work, a method based on the Minimum Description Length Principle (MDL) was applied. This was one of the first works to use 3D images in favor of the analysis of traffic signs [9].

In the work of Zhou and Deng [10], a system based on LIDAR (Light Detection and Ranging) and classification algorithms for the analysis of signaling plates images was used, using 3D data to improve the robustness of the task of detecting traffic signs. Through 3D point cloud data (color and spot clustering), the signboard in question was analyzed using Support Vector Machine (MVS) for classification of the traffic signs[10].

Another work, that also uses 3D data, was developed by Soiln et al. [11], each traffic sign was detected using the LIDAR sensing with classifiers based on semantic algorithms.

In the work of Wu et al. [12], a system that also uses LIDAR has been applied. This system does all the analysis through the 3D perception system. To make this possible, it uses landmarks to aid in the detection of signaling plates. [12].

3. Algorithm for detection of traffic signs with 3D data

Our algorithm for traffic signs detection uses a region of possible locations that plates usually can be found in the environment (Figure 1). In the urban transit environment, not always a traffic sign is placed on an individual pole, in some situations it can be found on a pole shared with other types of information (e.g. street name plates, light signals).

An Artificial Neural Network (ANN) with binary output has been trained with these various cases where boards (sign plates) and other elements can be found. The ANN was applied to solve this problem of classification and sign plate detection. For this to be possible, each type of case was modeled (Figure 1) based on the Velodyne LIDAR (Light Detection and Ranging) data and considering also a pair of stereo cameras, thus enabling the ANN network to respond if it is a board or an object that is not a sign plate.

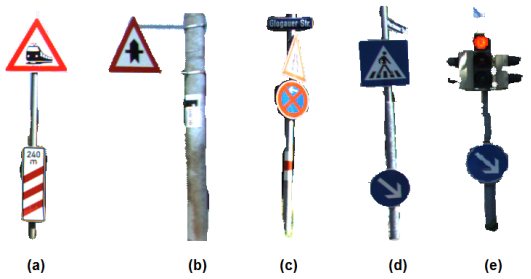

Figure 1. Traffic signs detected in different contexts (a) detected on a thin metallic pole (b) detected on a concrete pole (c) and (d) two sign plates detected on the same pole (e) traffic sign detected with a light signal

In case of the neural network algorithm informs the system that a board (traffic sign candidate) has been detected in the environment, then a second classifier based on Deep Learning CNN is activated to classify the type of traffic sign that was detected in image RGB-D (Red, green, blue + Depth): maximum speed, cones for route deviation, stop, preferential, pedestrian or also

other types of traffic signs.

In this situation, the 3D computer robotic vision system in conjunction with the ANN informs the detection and the coordinates where the possible traffic sign occurs (Figure 2). Given this information, the 2D image of the traffic sign detected is labeled and segmented, and submitted to the classifier based on the Deep Learning CNN, thus making it possible to identify which type of traffic sign was detected (Figure 2).

Figure 2. Detection of cones in a scene using 3D data - real scene with data provided by stereo vision camera system.

Deep Learnning CNN classification is applied on the 2D data (image-RGB), that takes advantage of the available textures and colors information to recognize the different specific traffic signs. The recognition power of Deep CNN Networks (DCNN-inception V3), is also a big advantage, once they can recognize and classify images very well (including images with occlusion, damaged and in precarious lighting conditions), and on the other hand, this task can be very difficult if considering only the 3D data (shapes-Depth).

In Figure 3, the 3D perception (visual attention) and computer vision system can be observed in action generating the point cloud of one scene using the stereo camera data from the KITTI Dataset. Still, it can be observed that the traffic sign and other objects were was detected with texture in the point cloud, these are some of the cases (Figure 3) trained in the ANN and used for detecting traffic signs objects.

Given Figure 3 it is possible to observe 2 types of segmented objects that are used to train our traffic signal detection system. Since one of them are not traffic signs (b) and the other one are sign used for stretches of the highway (a).

Each segmented object (Figure 3) will be sent to a 3D feature extraction method. After that, the data extracted from each object will be applied as the input

Figure 3. 3D-segmented objects (a) Cone - traffic sign, (b) Cyclist.

of an ANN-MLP for the classification of the detected object type.

By detecting a cone, our 3D vision system for auxiliary routes is activated, thus enabling the detection of an unmapped auxiliary route.

4. Theorical background and proposed model

4.1. Extraction of 3D features and classification of 3D objects

In order to recognize the traffic signs, it is necessary to first detect and segment 3D objects in the environment scene. Next, an ANN is used to recognize the point cloud structure representing a 3D signature of these segmented objects, indicating whether it represents a traffic sign or not.

To detect and recognize the traffic signal object, you must first extract some characteristics (features). This should allow the ANN classifier to have enough 3D data to declare whether a traffic sign object / structure was detected or not.

To generate a standard entry with recognizable patterns for the RNA, the so-called ”object signature”, 3D-Contour Sample Distances method [13] has been applied. The features extracted by the 3D-CSD are based on the principle that each object class has a unique 3D outline appearance. For this to be possible, the point cloud of the three-dimensional object is converted into a vector representing the surface contour distances related to a central point of the object, providing a 3D signature recognizable by Machine Learning techniques.

The 3D-CSD descriptor is applied by measuring the distances from the center of mass of the object to the specific surface points of the object (”external contour” or ”surface hull”), according to the selected

and predefined key points in a circumscribed sphere. More simply, the 3D object is placed within a virtual sphere which has a predefined number of scattered (usually equally spaced) points on its surface. By means of virtual rays, the measurements (center of mass of the object) are drawn from the center of the sphere towards each of these key points scattered over the surface of the sphere. Distances are measured from the center of the sphere to the first cross-point toward each key point (virtual radius length). The first intersection point could then be a point on the surface of the object (if the radius reaches the outer hull surface of the object), or a point on the surface of the sphere (if the radius does not reach the shell of the object), and is used for compose the vector of distances. This vector is used as the ANN input (input features vector).

The virtual rays applied to the sphere act as laser sensing rays, but in the other direction, instead of measuring the distance from the outer side to the surface of the object, the rays measure the distance from the center of the object to its surface. A video[1] was produced to explain this procedure. Thereafter, the measured distance values are interpolated to an appropriate (normalized) range and provided as input data from the machine learning classifier. We detail this procedure in the next section.

[1]

4.2. Point Cloud Segmentation

To apply the 3D-CSD descriptor, you must first segment objects in the point cloud environment scene. This is done using clustering algorithms, such as k-means or a simple background subtraction in some cases. Each segmented object in the scene is processed in an individual way and allows the generation of 3D-CSD descriptors and followed by pattern recognition.

4.3. 3D-CSD Feature Extraction

The first step is to estimate a surface mesh for the point cloud of the object by converting the sparse point cloud into a set of polygons (surface mesh) and obtaining the solid representation of the 3D shape object. We use the 3D Convex Hull algorithm for this task, so that the approximate contour of the object (shell of the object) can be obtained.

Next step, consider a sphere circumscribing the 3D Convex Hull (Figure 4). The spheres center is positioned at the objects center of mass, and the radius is the furthest objects point from center. Figure 5 illustrates this process.

The next step is to select the various key points on the sphere. Each key point will correspond to a single distance measure and therefore a position in the vector of final distances. The key points should be equally distributed or, as a priority, the most representative areas of objects. For traffic signals, a generic strategy for distributing points along the surface of the sphere can be performed by defining an azimuth angle and constant altitude for the point distribution (Figure 4).

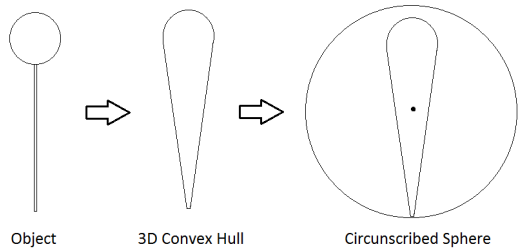

Figure 4. 2D view of sphere generation process.

After selecting the points, the 3D-CSD (object distance vector) descriptor can be generated. For each key point p_i on the surface of the sphere, it is then considered a straight line from the center of p_i for the sphere. If this straight line crosses the surface of the object at a point d_i, calculate the Euclidean distance from point d_i to the center of mass of the object. If not, return -1. This process must always be done sequentially, for example, from left to right and then from the top to bottom sequence. The figure 5 illustrates the procedure for measuring distances.

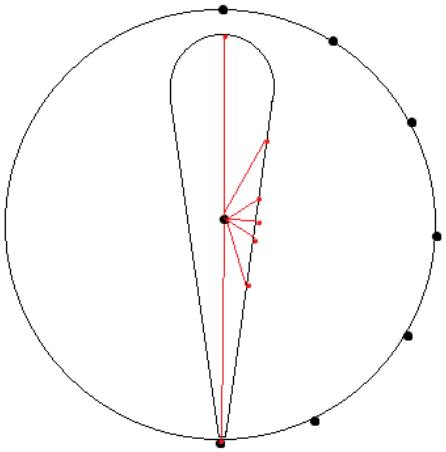

Figure 5. Example of distances measurement of object.

After all distance measurements are made by

[1] 3D-CSD Feature Extraction explained in vdeo Sphere, Virtual Rays and the Vector of Distances concepts Available at: https://goo.gl/K5X5yB

3D-CSD, the values must be interpolated (that is, normalized) ensuring adequate input to the ANN. Another strong point is that this interpolation procedure is the key to providing scale invariance since the shape aspect is estimated based on distance proportions rather than metric measures. Our descriptor values must be in the range [0..1] , where 1 is the radius of the sphere.

4.4. Pattern Recognition

In order to be able to recognize the 3D signature of the segmented object given by the 3D-CSD feature, an ANN Multi-Layer Perceptron (MLP) is used because of its good capabilities for capturing complex, non-linear underlying features with a high degree of accuracy. Figure 6 shows the ANN architecture.

The ANN input layer must contain the same number of units as the vector size of the distances and contours that are normalized. The ANN output layer is composed of a single unit, for a binary output (signal object (1) / non-signal (0)). The size of the hidden layer of the ANN was defined empirically, making it possible to guarantee the minimum number of units required for a good convergence and generalization of learning.

The supervised learning method of ANN requires a set of training data that must be generated a priori. It is necessary to assemble a representative set of examples, composed of a large set of objects from different scenes, applying the descriptor 3D-CSD and labeling each example as a signal / non-signal object. In order to be able to recognize the objects from different points of views (different orientations), additional examples of objects in different rotations are included in the training data set.

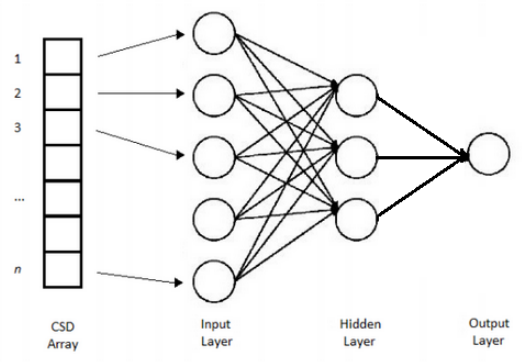

Figure 6. ANN architecture.

5. Deep learning CNN to classify the type of traffic sign

In the system presented in this paper, the 2D images representing the traffic sign are sent to a classifier based on a Deep Learning model, adopting a DCNN (Deep Convolutional Neural Network / ConvNet) network implemented using TensorFlow [14].

We used the free software library for machine learning TensorFlow [14]. The CNN networks have been successfully applied to image recognition of different classes, from manuscript digits (MNIST) to images of objects / animals (ImageNet Large Scale Visual Recognition Challenge (ILSVRC)), and is now being used in many real world applications. TensorFlow was developed for machine learning and research in Deep ANNs [14].

The structure used by Google in building its Deep Convolutional Neural Network (DCNN), known as GoogLeNet or Inception, has achieved excellent results in the ImageNet competition. The Inception network, and the new versions of it, with the adjusted ANN weights previously trained to the ImageNet dataset, are available in Tensorflow. It works with a large number of hidden layers, which is what defines this type of network as a "deep neural network". For our application we apply transfer learning in the network, and we get good results. Through transfer learning we inserted 21 new classes of traffic signs into the network. We do not modify the structure, so we take advantage of all filters and convolutions [14].

By means of the Figure 7 it is possible to observe a situation where our where our 3D object detection method recognized a cone. After that, we send the image (square in red) to the network of Deep Learning to classify which is the signal of traffic that was detected.

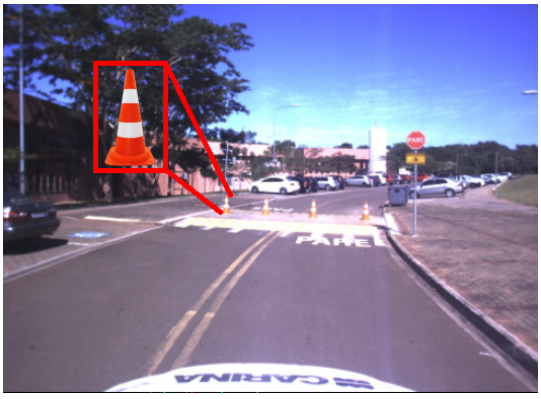

Figure 7. Cone object classification with Deep Learning - TensorFlow.

5.1. Methodology: Training and Testing

For the DCNN training, we used the data set called the German traffic signal data set (INI) [15]. However, for the system tests, we applied another database. We used the scenes from the KITTI Vision Benchmark Suite (KITTI) [16]. In these scenes [16], our 3D traffic signal detection system was applied and then the 2D candidate images retrieved from traffic sign objects are sent to the Deep Learning classifier. The use of two databases enabled the validation of the system, since it was tested with a data set completely different from the one used in the training.

The training based on transfer learning applied to CNN-Inception modified only the final layer of the network [17], guaranteeing the preservation of all the filters and convolutions that are generic for the recognition of different types of images based on textures, colors, and other types of patterns [14]. In order to be able to adapt the network, 70% of the data set was applied to perform the training in the network (adapting the network parameters - adjusting the weights). The remaining 30% was applied only to evaluate the performance of the network in the test steps (validation) in the image recognition task [18].

To avoid overtraining / overfit,and to improve learning, larger training data sets should be used, with a cross-validation training scheme, and also using a good detection method to extract the background of the image of interest (better image segmentation).

6. Grid 3D Map

Since we have a map, in this context of "smart cities" integrated to "autonomous vehicles", the navigation system can easily plan a trajectory and pass it to the vehicle. Nevertheless, in the scope of this work we focus on specific cases where the mapped lane can be obstructed and an auxiliary route is given, to keep the vehicles navigating in that region. In this situation, the alternative route is not mapped, the vehicle needs then to identify the obstructed road and look for an alternative navigable area so it can reach the destination.

We propose a Stereo based grid map fused with visual odometry. Using Stereo data and ORB-SLAM localization method we can concatenate each stereo frame point cloud, aggregating information in each new mapped region.

6.1. Visual Localization

Since we use stereo images to generate the 3D points of the environment, to recreate the trajectory in a point cloud, we need a reliable localization and pose estimation of the vehicle in each camera frame. Some sensors like GPS and IMU give a precise vehicle global position, the problem in this case is the synchronization of the data from multiple sensors. This could aggregate an incremental noise that would spoil the concatenation of the points and the final map.

Figure 8. (a) The ORB-SLAM system extracted features in green; (b) The ORB-SLAM features mapped in red points, in blue the keyframes pose and in green the estimated trajectory.

Thereat we used the ORB-SLAM [8], it is based on the ORB (Oriented FAST and Rotated BRIEF) feature extractor. This feature detector is a fusion of FAST [19] keypoint detector and BRIEF [20] descriptor with modification for performance optimizations. The system proposes a visual odometry method that is based on image feature extraction robust to both transformations, translation and rotation.

During the trajectory keyframes are selected to map the environment, features are extracted and mapped in the stereo-pair images with its 3D position, so new frames poses can be estimated using the mapped keyframes. The first frame is considered as the origin of the system and new frames have the pose calculated as a transformation from the origin to its estimated position. Correspondence features in left and right images are mapped to support the pose estimation whenever this item is found. Based on the depth information the features are divided between two classes, $close$ and far. Points closer to the camera have more reliable information about translations, and further points, reliable data about rotations. This knowledge about the depth of the points is important, based on it we can have a more precise pose estimation of the vehicle. As the vehicle navigates through the environment new characteristics are extracted and saved. Features already mapped are used to estimate the pose of the actual frame in the map. In the Figure 9 we have a comparison between the ORB-SLAM localization based on the stereo images and the satellite Google Maps view of the executed trajectory.

With the pose at each frame, we can generate and join the 3D points in a complete trajectory point cloud, keeping the main characteristics in each mapped region.

Figure 9. ORB-SLAM trajectory points in red overlaid in Google Maps view of the trajectory.

6.2. Grid Semantic Mapping

The problem with the concatenation of point clouds is the number of points stored. Since the stereo images generate a dense depth image, in the concatenation all those points are joined and a massive volume of data have to be processed in order to use this map. To solve this we used a Grid Map approach. For each grid we have a block size where any points falling in this range will be considered as a point belonging to this grid block.

As we have an agglomeration of points, a grid division can reduce the number data and characterize a specific region in the trajectory based on all the information gathered in the grid. The division is based on a fixed size where we divide the width and the depth range of the trajectory by this b block size parameter. For each point in the concatenated point cloud we calculate the grid block which it will fall and associate its height to the grid. When all the points are fitted, we use the heights stored to estimate the mean height for each grid block. The density of the concatenated point cloud guarantee a reliable representation of a region mapped in a block range since we have a size b small enough.

To extract these associated height characteristics we use a simple statistic calculation. For each grid block depth t we calculate the mean and standard deviation height of the actual t, the previous $t-1$ and the next $t+1$ grid blocks depth. The mean $\overline{B_t}$ and the standard deviation S_t are calculated as follows, where n is the number of width blocks in a depth t:

$$\overline{B_t} = \frac{\sum_{i=t-1}^{t+1}\sum_{j=1}^{n} B_{i,j}}{3n} \tag{1}$$

$$S_t = \sqrt{\frac{\sum_{i=t-1}^{t+1}\sum_{j=1}^{n}(B_{i,j} - \overline{B_t})^2}{3n-1}} \tag{2}$$

With these values we set the points that are higher than the threshold $T_t = \overline{B_t} + S_t$ as positive obstacles and marked in red, then we recalculate the mean and standard deviation for the remaining blocks. With the new mean $\overline{B_t^n}$ and standard deviation S_t^n we classify any points above an upper threshold $U_t = \overline{B_t^n} + 1.5S_t^n$ and a lower threshold $L_t = \overline{B_t^n} - 2S_t^n$. Any points above U_t are also considered as positive obstacles, points under the L_t threshold are considered negative obstacles and marked in green, the remaining points are traversable area, marked in light blue. It is important to note that this approach is not robust for environment variation since the mean and standard deviance vary according to the obstacles heights in the environment, but for our case these values could classify the mapped points as we can see in the Figure 10.

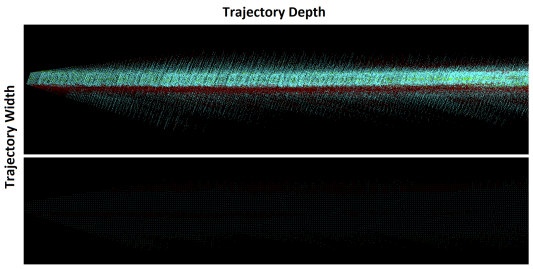

Figure 10. In the above image we have the trajectory dense concatenated point cloud. In the image below, our grid point cloud approach. Red are the positive obstacles, in green the negative obstacles and in light blue the traversable area.

With a grid point cloud (Figure 10), each point represents the data that have been joined in that region. Thereby we have few information stored in the resultant map, but each point have an important and reliable information associated, based on the points close to each the grid block center. Also, with the semantic information about obstacles aggregated in each grid block, the vehicle can identify where it is able to navigate.

7. Experiments and results

We apply our 3D traffic sign detection method to track traffic sign objects and their images from the

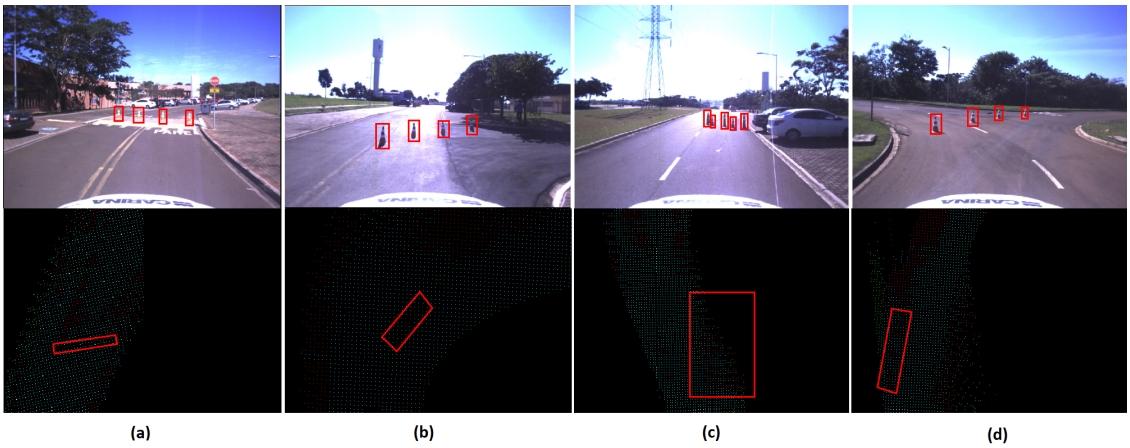

Figure 11. Scenes with image cone detection and respective grid map with cones mapped as obstacles and the alternative route mapped as traversable area.

KITTI dataset and also in our dataset with intermediate sections that use cones for traffic signaling.

Our vision system was also able to generate unmapped help routes based on 3D vision. Enabling greater robustness related to route replanning in situational situations, road accidents or even situations where a secondary road (rural roads, auxiliary roads) is not on the current map of the autonomous vehicle controller.

7.1. Detection of traffic signs using 3D data

Given the results of the classification of 3D objects obtained by ANN with binary output, it was possible to state whether the detected object is a traffic sign or not. If the object was a traffic signal, the Deep Learning based system is enabled to evaluate the 2D image, recognizing what type of traffic signal was detected. We obtained a hit rate of about 83 % for the detection task and 98.2 % for the recognition of the different traffic sign images through the Deep Learning network.

Our detection error is related to false positives (9 %) and false negatives (8 %). False positives and false negatives are related to objects that are very similar to the shape of the objects that were applied in the training of the Neural Network. These objects are related to trees, people and cars. We chose these objects because they are easier to confuse with traffic sign objects (Figure 12), given their somewhat similar formats.

7.2. Grid Vision System

Based on the reliable pose estimation from the ORB-SLAM system we can correctly concatenate each point cloud. The main problem rely on keeping the distribution from the original points in the new grid map. The ranges of the input point clouds used in our experiments are shown in Table 1, also we used $b = 0.1$ for the block size parameter.

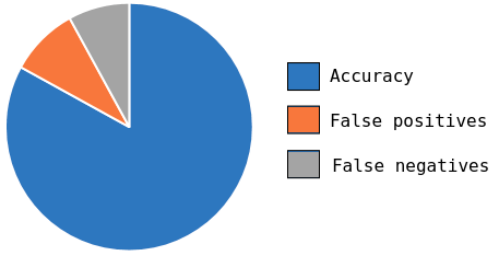

Figure 12. Accuracy of the traffic sign detection system.

Table 1. Input Point Cloud Ranges.

	Width	Height	Depth
Min	0.6	0.2	0.4
Max	7.0	2.0	10.0

To evaluate this distribution we calculate the mean and standard deviation of the original concatenated point cloud and the generated grid point cloud. In Table 2 we can see that the mean values from the grid method stay near to the original distribution. The difference in x and z variance is due to the fact that in the original point cloud we have an accumulation of information in the center of the trajectory, while in the grid all the points are equally distributed in the x and z ranges. Also, the variance in the y we have the height distribution that stay real close to the original point cloud. These results show that we have equally allocated points in the width and depth ranges and maintain the height distribution. Also, we could reduce the volume of points approximately

in 96%, storing fewer data and keeping the original distribution.

In Figure 11 the cones are detected in our 3D and 2D data fusion method. Under the image detection, we have the grid map generated with the trajectories. Cones are identified and mapped as positive obstacles as marked in the red boxes, and the auxiliaries routes are mapped as traversable area. This results shows that with the road obstruction detected, our mapping system is able to identify the blockage and find a possible traversable area that can be used as an auxiliary route to the vehicle destination.

8. Conclusions and future work

In this work, we developed a new system capable of detecting and classifying traffic signals and cones through the fusion of 2D and 3D data, as well as detecting common routes in sections of interdicted highways. We achieve this through a semantic analysis of objects declared as traffic signs. Our system is capable of detecting and classifying diverse types of traffic signals and, when a cone or a diversion signal is detected, our 3D vision system is able to find an auxiliary route.

Analysing the available 3D data it was possible to detect the traffic signals and cones using a good descriptor in conjunction with an ANN, thus obtaining a greater robustness. The 2D data enabled an efficient classification, thanks to the ability of Deep Learning and its power over images with texture, colors and shapes.

In future works we plan to develop a new and more robust method for obstacle detection in the point cloud, grid mapping with more semantic information about the obstacles, giving to the vehicle a more reliable and detailed information about the environment.

We also intend to apply this system in practical situations and intelligent/autonomous vehicles. We are working in adapting it to our own vehicles, developed in the LRM Lab., the first autonomous truck of Latin American (Figure 13 (a)) and also in CARINA 2 Project (Intelligent Robotic Car for Autonomous Navigation Project) (Figure 13 (b)). Then testing it in real environments with images obtained in real time.

9. Acknowledgement

We thank CAPES and CNPq for the support to Diego Renan Bruno, Lucas P. Nunes Matias and Jean Amaro of graduate program of University of São Paulo at São Carlos.

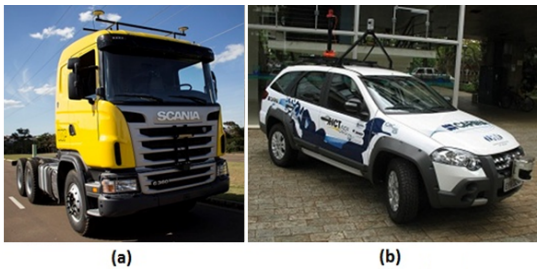

Figure 13. Autonomous vehicles of the LRM Lab. / USP. (a) First autonomous truck in Latin America; (a) The CaRINA 2 Project (Intelligent Robotic Car for Autonomous Navigation project)[21].

References

[1] K. Stiens, G. Tanzmeister, and D. Wollherr, "Local elevation mapping based on low mounted lidar sensors with narrow vertical field of view," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 616–621, Nov 2016.

[2] K. Stiens, J. Keilhacker, G. Tanzmeister, and D. Wollherr, "Local elevation mapping for automated vehicles using lidar ray geometry and particle filters," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, pp. 481–486, June 2017.

[3] G. Reina, A. Milella, and R. Worst, "Lidar and stereo combination for traversability assessment of off-road robotic vehicles," *Robotica*, vol. 34, no. 12, p. 28232841, 2016.

[4] M. Bajracharya, J. Ma, M. Malchano, A. Perkins, A. A. Rizzi, and L. Matthies, "High fidelity day/night stereo mapping with vegetation and negative obstacle detection for vision-in-the-loop walking," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3663–3670, Nov 2013.

[5] A. Romanoni, D. Fiorenti, and M. Matteucci, "Mesh-based 3d textured urban mapping," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3460–3466, Sept 2017.

[6] S. Sengupta, E. Greveson, A. Shahrokni, and P. H. S. Torr, "Urban 3d semantic modelling using stereo vision," in *2013 IEEE International Conference on Robotics and Automation*, pp. 580–585, May 2013.

[7] J. Engel, J. Stckler, and D. Cremers, "Large-scale direct slam with stereo cameras," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1935–1942, Sept 2015.

[8] R. Mur-Artal and J. D. Tards, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE Transactions on Robotics*, vol. 33, pp. 1255–1262, Oct 2017.

[9] R. Timofte, K. Zimmermann, and L. Van Gool, "Multi-view traffic sign detection, recognition, and 3d localisation," *Mach. Vision Appl.*, vol. 25, pp. 633–647, Apr. 2014.

[10] L. Zhou and Z. Deng, "Lidar and vision-based real-time traffic sign detection and recognition algorithm for intelligent vehicle," in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pp. 578–583, Oct 2014.

Table 2. Mean, standard deviation and number of points from original concatenated point cloud and the generated grid point cloud.

	Full Point Cloud			Grid Point Cloud		
	Mean	Variance	Points	Mean	Variance	Points
X	1.9261	11.0822	424817	3.2194	21.0107	14923
Y	0.2347	0.0033	424817	0.2243	0.0049	14923
Z	-23.1317	97.4283	424817	-24.7042	110.3002	14923

[11] M. Soiln, B. Riveiro, J. Martnez-Snchez, and P. Arias, "Traffic sign detection in mls acquired point clouds for geometric and image-based semantic inventory," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 114, pp. 92 – 101, 2016.

[12] S. Wu, C. Wen, H. Luo, Y. Chen, C. Wang, and J. Li, "Using mobile lidar point clouds for traffic sign detection and sign visibility estimation," in *2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pp. 565–568, July 2015.

[13] D. O. Sales, J. Amaro, and F. S. Osrio, "3d shape descriptor for objects recognition," in *2017 Latin American Robotics Symposium (LARS) and 2017 Brazilian Symposium on Robotics (SBR)*, pp. 1–6, Nov 2017.

[14] D. R. Bruno and F. S. Osorio, "Image classification system based on deep learning applied to the recognition of traffic signs for intelligent robotic vehicle navigation purposes," in *2017 Latin American Robotics Symposium (LARS) and 2017 Brazilian Symposium on Robotics (SBR)*, pp. 1–6, Nov 2017.

[15] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "The German Traffic Sign Recognition Benchmark: A multi-class classification competition," in *IEEE International Joint Conference on Neural Networks*, pp. 1453–1460, 2011.

[16] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.

[17] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *CVPR*, pp. 2818–2826, IEEE Computer Society, 2016.

[18] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "Decaf: A deep convolutional activation feature for generic visual recognition," in *Proceedings of the 31st International Conference on Machine Learning* (E. P. Xing and T. Jebara, eds.), vol. 32 of *Proceedings of Machine Learning Research*, (Bejing, China), pp. 647–655, PMLR, 22–24 Jun 2014.

[19] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Computer Vision – ECCV 2006* (A. Leonardis, H. Bischof, and A. Pinz, eds.), (Berlin, Heidelberg), pp. 430–443, Springer Berlin Heidelberg, 2006.

[20] M. Calonder, V. Lepetit, M. Ozuysal, T. Trzcinski, C. Strecha, and P. Fua, "Brief: Computing a local binary descriptor very fast," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, pp. 1281–1298, July 2012.

[21] "CaRINA 2 Project." `http://lrm.icmc.usp.br/web/index.php?n=Port.ProjCarina2Info`. Accessed: 2018-06-12.